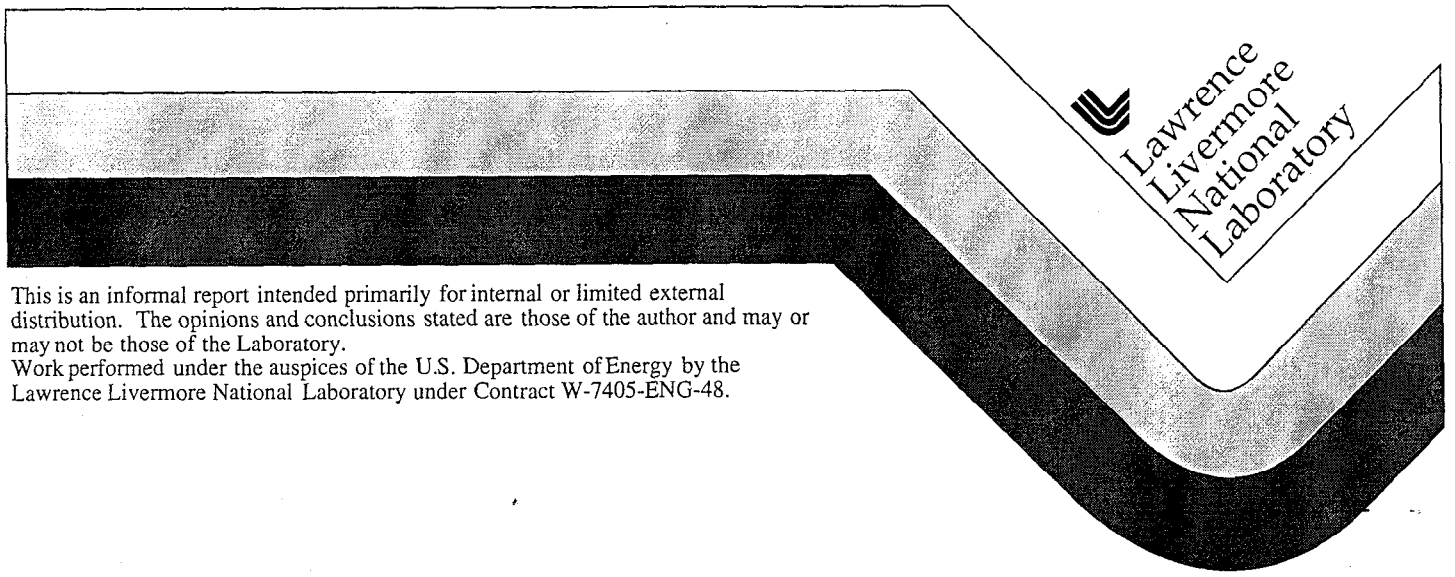
UCRL-ID-133147

Integrated Computer Control System Status Monitor Simulation

Brett M. Kettering

September 18, 1998



Integrated Computer Control System Status Monitor Simulation

FY98 LDRD Report

September 18, 1998

Brett M. Kettering

Introduction

Simulations of the ICCS (Integrated Computer Control System) status monitor framework have been performed. The results provide confidence that the requirements related to the framework can be met. The simulation was done using the SIMPROCESS discrete-event-modeling tool. This document describes the results of the simulations.

The requirements related to the status monitor framework are:

> *1 graphical user interface (GUI) per workstation supports 10 status updates per second.*
>
> *Broad-view control system status updates can be received in less than 10 seconds in the expected configuration of the ICCS for NIF deployment.*

Overview

ICCS will run under both the Solaris and VxWorks operating systems. In most cases the server-side status monitor will be an integral part of an FEP application and run on the VxWorks operating systems, though several FEP applications do run on the Solaris operating system. Most client-side status monitor implementations will be integral parts of ICCS supervisory applications and run on the Solaris operating system-based consoles, though some supervisory applications do not run on consoles. Figure 1 graphically depicts the ICCS network that will use the status monitor framework software.

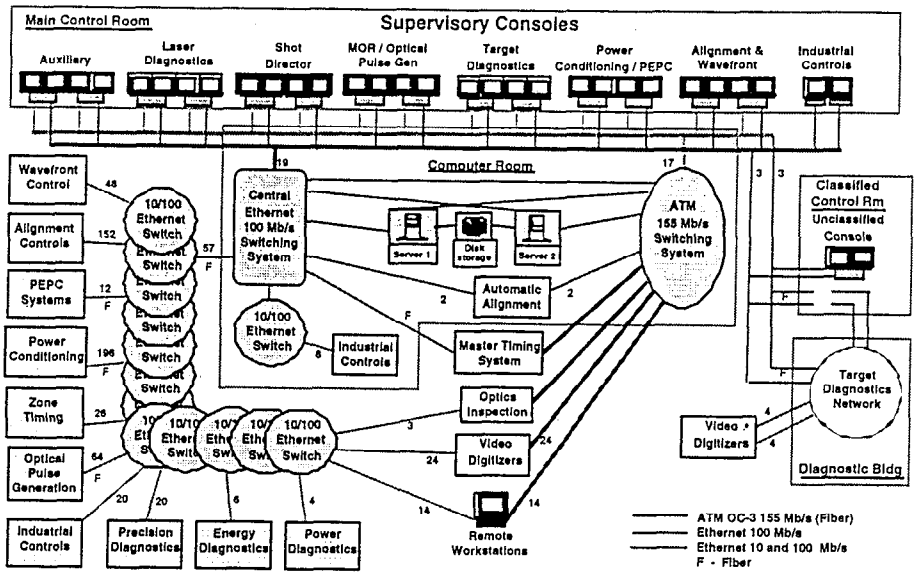


Figure 1 – ICCS Network Configuration

The simul s were setup to emulate three expected FEP types that use the 10Hz, 1/1(and 1Hz polling rates. The results are reported in this document mmendations on how to alter the simulations to include even more rea usage patterns are provided.

Assum ns of the Model

Discrete t simulations present some limitations in modeling complex, non-det listic behaviors of time-sharing, multi-tasking computers and their o ng systems. Some assumptions were made to simplify the model yet stil tain confidence that the result is reasonable. This section descri ose assumptions.

Work f FEP

The si tions were performed with the total resources of the server-side and c -side application computers available at all times. The status monit pplications did not contend with other applications or operating syste or resources during the simulations.

Polli arameters of the Models

A se r-side status monitor application was written that could be configured for p l rate, whether or not to report an update at each poll, and the number of de ice attributes to poll.

A client-side status monitor application was written that could be configured for the poll rate of the server-side application, whether or not updates would be reported at each poll, and the number of device attributes to be polled. By this manner it could count the number of updates received versus the number expected.

To obtain a benchmark of capacity for integer updates these applications were configured in different manners and experiments run. The computers were not doing any other task at the time such that the only resource usage would come from the operating system and the applications themselves. A graphical representation of the results is shown in Figure 2.

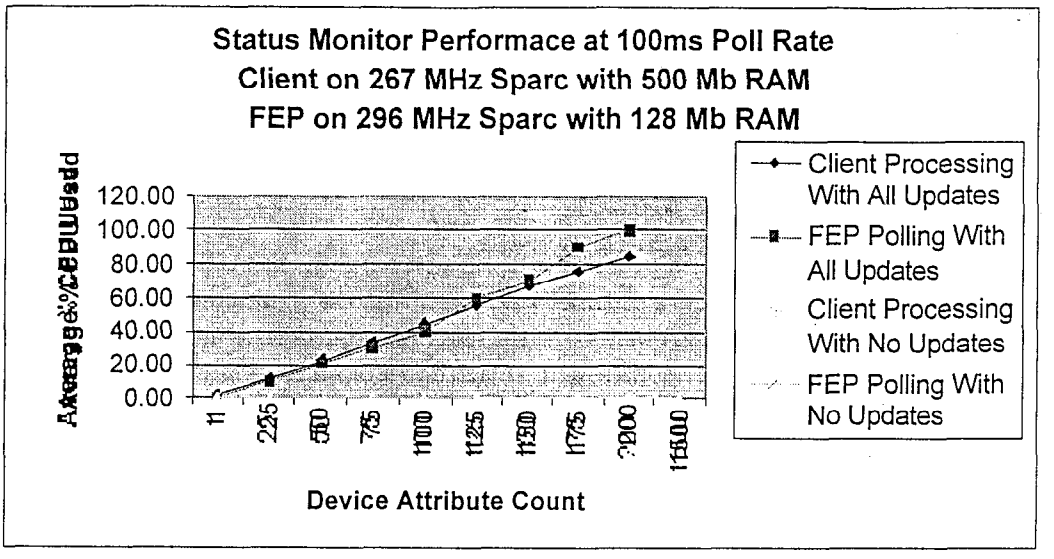


Figure 2 – Capacity Benchmark

The first experiment was set for a 10Hz poll rate and not to report any updates. The device attribute count was set at several values and the CPU load was measured. The client-side application computer registered almost no load. This was expected, as the application itself was not doing any work. The server-side application computer gradually increased in load until it was saturated at 1,500 device attributes. Since 1,500 device attributes could be polled 10 times per second without generating any updates it was concluded that the cost per device attribute poll was approximately 67 microseconds. This is for a device attribute whose value is maintained in the server-side application computer's memory.

The FEP engineers estimated that 1 millisecond would be a reasonable time to poll a device attribute value over an ICCS FEP bus. This was the value used in the simulations.

The second experiment was set for a 10Hz poll rate and to report an integer update for each poll. The device attribute count was set at several values and the CPU load was measured. The results show that the usage increased linearly with the number of devices polled until the server-side application computer was saturated at 200 device attributes. It was concluded that the cost per update sent was approximately 500 microseconds.

The CORBA Fast-Track project performed a much more extensive set of experiments with various sizes and types of data. It was shown that the time to package and send CORBA updates varied with the size and type of data being sent. The simulation values used for the CORBA packaging and sending

were based on the CORBA Fast-Track project results and the expected sizes and quantities of data for the types of FEPs modeled in the simulations. The values were somewhat greater than 500 microseconds.

Time of Task Parameters of the Models

Experienced control system designers estimated 5 milliseconds work for updating GUI when a device attribute value was received. This was the valued used in the simulations.

Device Attribute Counts and Use of Beta Statistical Distribution

The simulation passes entities (e.g. updates) through a sequence of activities (e.g. polling and comparing, packaging to send, etc.). Each activity has, among its other attributes, a delay. This delay defines how long it takes an entity to finish the activity. The sum of these delays from beginning to end for all entities is the total time required to complete the process modeled by the simulation. These delays can be constants or random numbers chosen from a statistical distribution.

Each scenario had a minimum and maximum number of updates that could occur at each poll. The Beta distribution allows a minimum and maximum bound to be put on the distribution and a bell shape to be aligned where desired within those bounds. Thus, the Beta distribution was used to generate random number of updates at each poll in the startup simulations.

The Beta distributions were established based on expected ICCS FEP device attribute counts in the FEPs that would use the simulated poll rates and the experience of ICCS team members with computer systems.

Description of Status Monitor Activities

A SIMPROCESS *activity* is a task, through which the simulation entity must pass and perform some work. A SIMPROCESS *connection* is a line of flow from one *activity* to another. This section describes each of the simulation activities used in the status monitor scenarios.

Update Pack Per Client

A random number of updates are generated at each poll rate. The range of update count possibilities is different for each poll rate in the various experiments. This activity groups all the updates into a single packet with an indication of the number of updates in the pack. This is done because the status monitor software sends all updates to a single client application in a single message, which is then unpacked by the client.

FEP Polling and Comparing

This activity calculates the time to poll and compare the number of updates that were generated. This value is based on the 1-millisecond estimate of the FEP engineers.

Packaging for Send

Once the updates had been polled and compared they are packaged in a CORBA message to be sent to the client. This activity calculates that time based on the numbers obtained from the CORBA Fast-Track project.

FEP CORBA Send

Once packaged the update is sent to the client and received by it. This activity calculates that time based on the numbers obtained from the CORBA Fast-Track project.

Separate Update Pack

This activity does not add time to the simulation. It takes the packet of updates and introduces that many entities into the simulation to be processed individually by the client application.

Client Processing

The client application processes each application, which commonly results in the update of a GUI on the operator console. The value per update is based on the 5-millisecond estimate of the control system designers.

Results of the Simulation

Three simulations were performed based on expected ICCS FEP configurations. They were the Alignment Controls FEP at 10Hz, the Industrial Controls FEP at 1/10Hz, and the PEPC LRU FEP at 1Hz.

Alignment Controls FEP

The Alignment Controls FEP monitors a manually controlled device at 10Hz during its use. Each device may consist of zero to eight motors. Thus, each 10Hz poll yields zero to eight updates with bias toward the lower quarter. That is, it is not common that all eight motors move at the same time. Figure 3 shows the graph of the PDF (Probability Distribution Function) and the CDF (Cumulative Distribution Function) used to generate the random number of updates at each poll cycle.

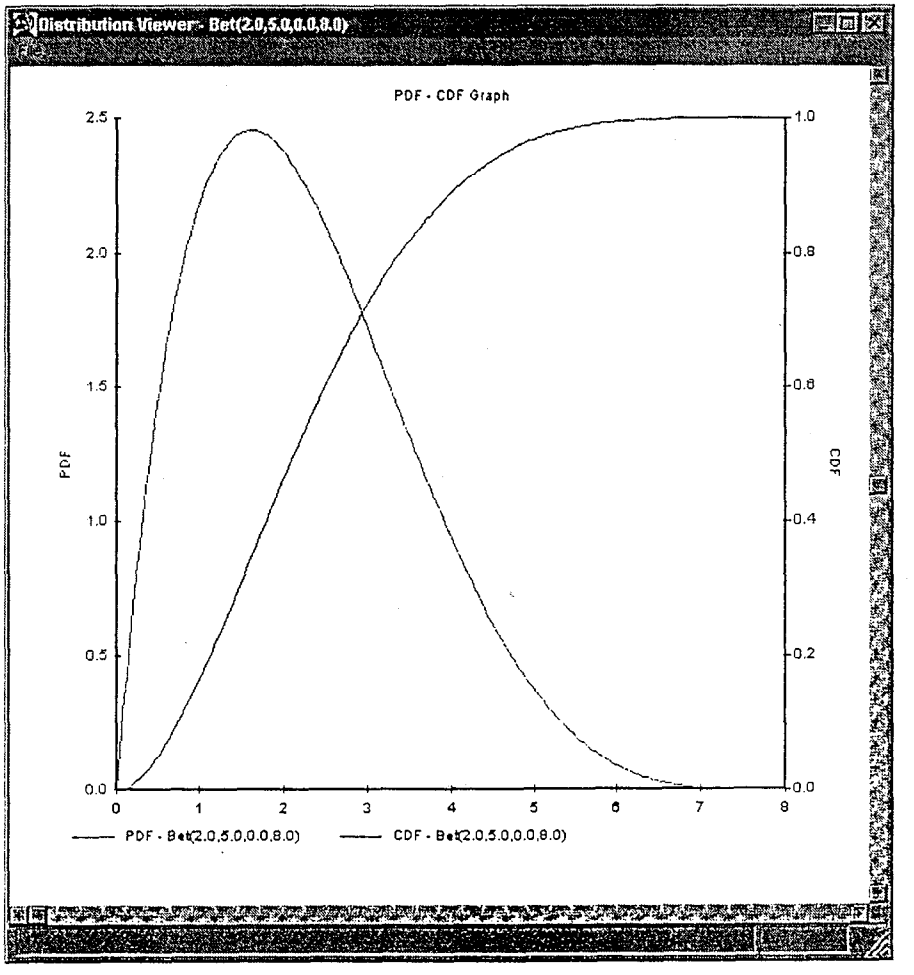


Figure 3 – 10Hz Random Distribution Functions

The simulation of one Alignment Controls FEP updating one workstation for 30 minutes showed that the 10Hz rate could be maintained without any updates being processed late. Figure 4 graphically depicts the results of this simulation.

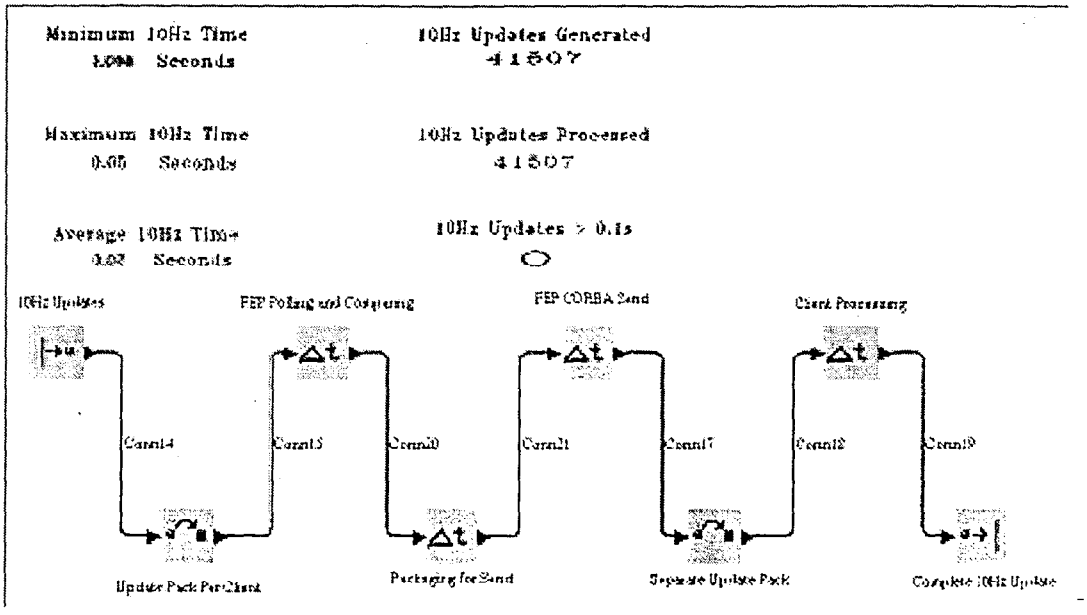


Figure 4 – One Alignment Controls FEP to One Workstation

The simulation of one Alignment Controls FEP updating two workstations for 30 minutes showed that the 10Hz rate could be maintained without any updates being processed late. Figure 5 graphically depicts the results of this simulation.

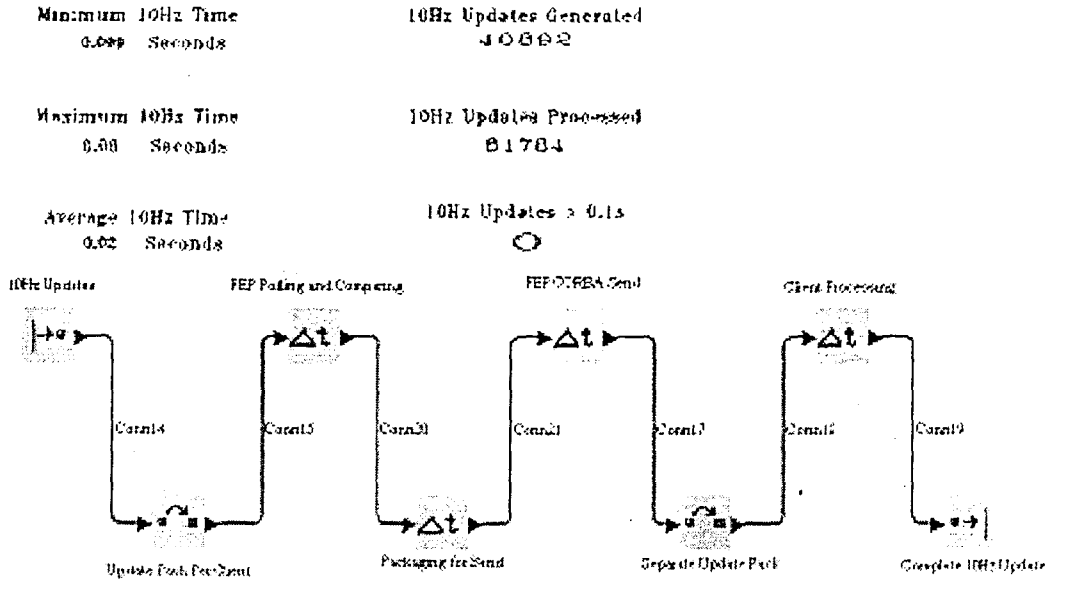


Figure 5 – One Alignment Controls FEP to Two Workstations

One Alignment Controls FEP updating 14 workstations for 30 minutes quickly fell behind and showed many updates arriving late. It is expected that only one operator will control any one device at any given time. Perhaps, two operators will control two devices controlled by the same FEP, but it is unlikely that 14 operators will be controlling 14 devices controlled by the same FEP. Thus it is concluded that the status monitor can meet the requirement of updating one GUI per workstation at 10Hz, especially where the sources of the updates are from independent FEPs.

Industrial Controls FEP

The Industrial Controls FEP monitors general status of many sensors at 1/10Hz. On average an Industrial Controls FEP monitors 272 sensors. Thus, each 1/10Hz poll yields zero to 272 updates with bias toward the lower quarter. That is, it is not common that all 272 sensors update at the same time. Figure 6 shows the graph of the PDF (Probability Distribution Function) and the CDF (Cumulative Distribution Function) used to generate the random number of updates at each poll cycle.

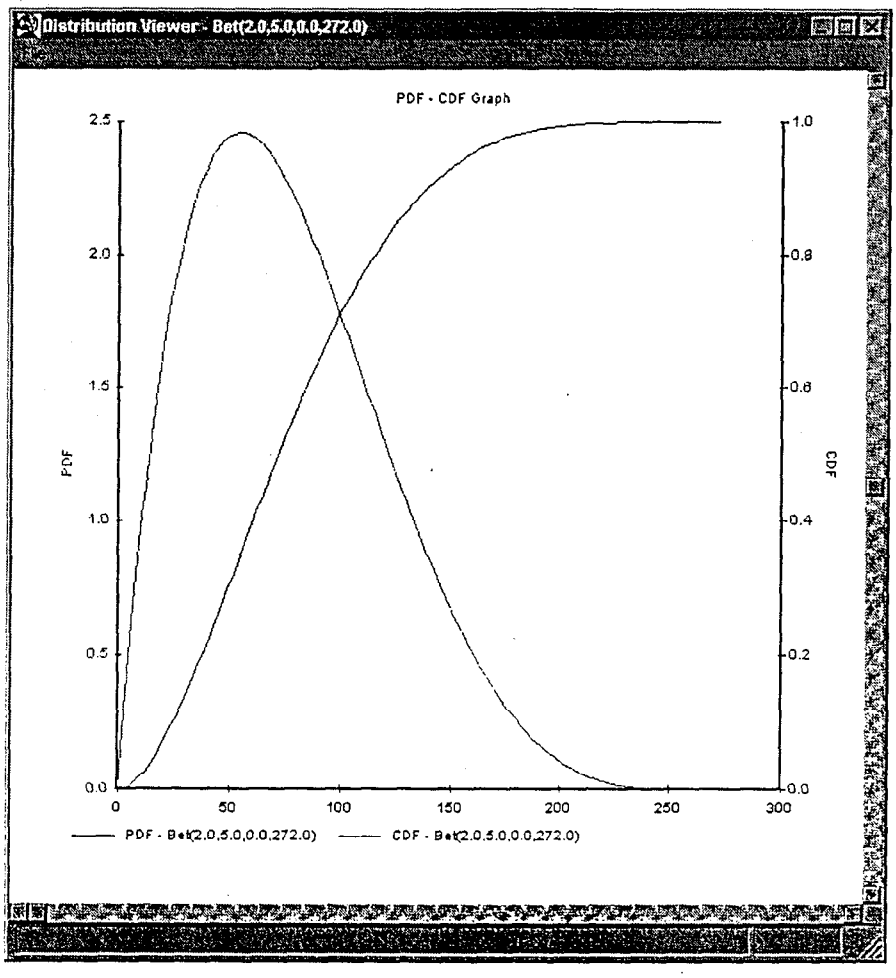


Figure 6 – 1/10Hz Random Distribution Functions

The simulation of one Industrial Controls FEP updating one workstation for 30 minutes showed that the 1/10Hz rate could be easily maintained without any updates being processed late. Figure 7 graphically depicts the results of this simulation.

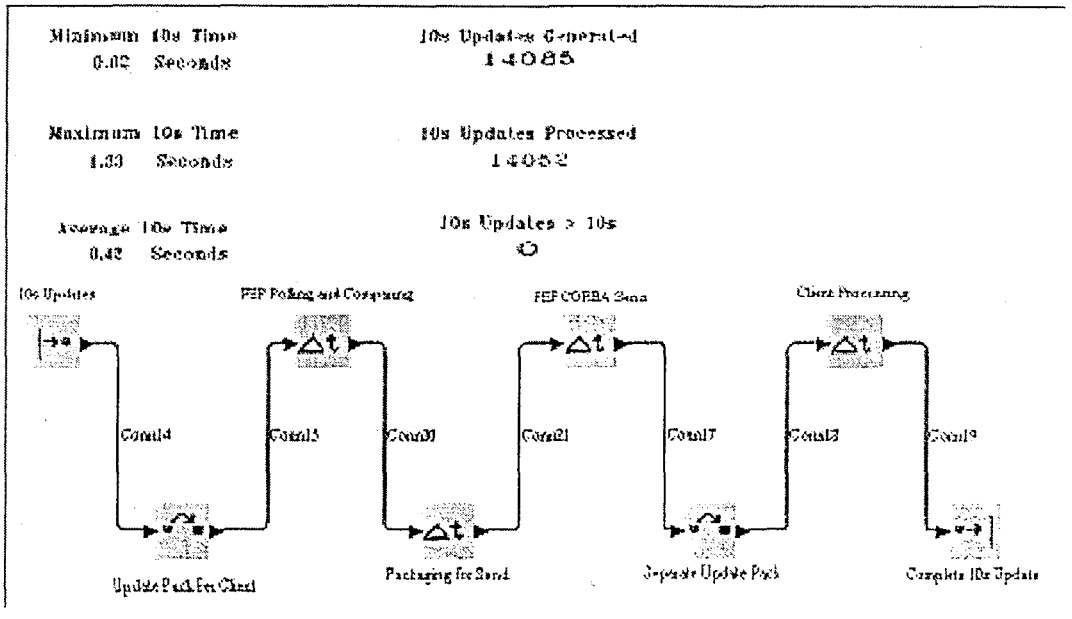


Figure 7 – One Industrial Controls FEP to One Workstation

The simulation of one Industrial Controls FEP updating 14 workstations for 30 minutes showed that the 1/10Hz rate could be easily maintained without any updates being processed late. Figure 8 graphically depicts the results of this simulation.

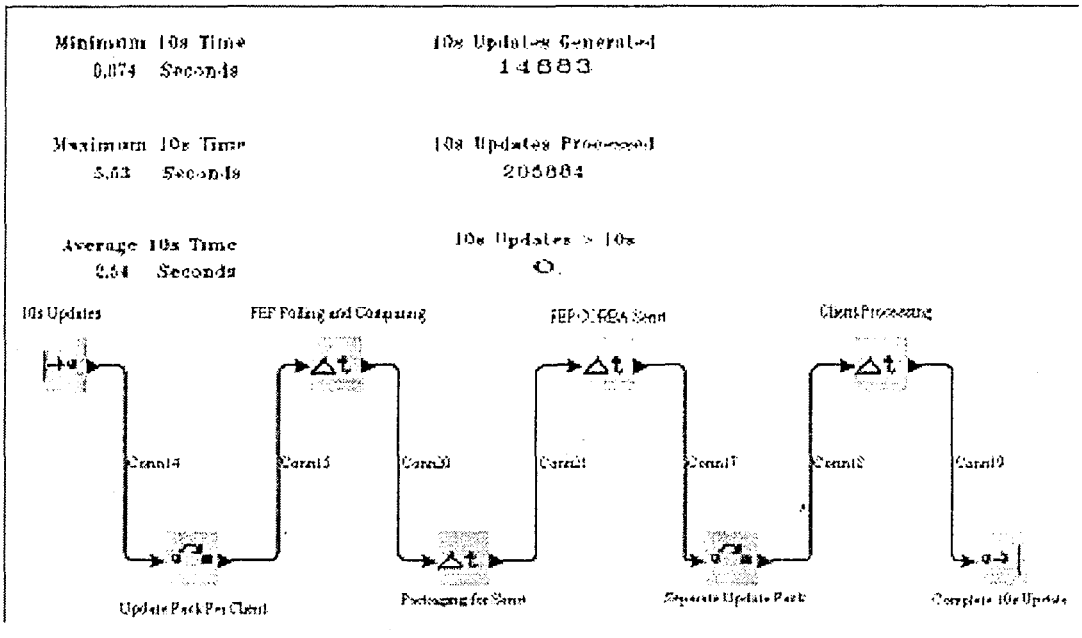


Figure 8 – One Industrial Controls FEP to 14 Workstations

Thus it is concluded that the status monitor can meet the requirement of broad-view control system status updates at 1/10Hz.

PEPC LRU FEP

The PEPC (Plasma Electrode Pockels Cell) LRU (Line Replaceable Unit) FEP monitors seven devices on each of 12 LRUs at 1Hz. Thus, each 1Hz poll yields zero to 84 updates with bias toward the lower third. That is, it is not common that all 84 devices update at the same time. Figure 9 shows the graph of the PDF (Probability Distribution Function) and the CDF (Cumulative Distribution Function) used to generate the random number of updates at each poll cycle.

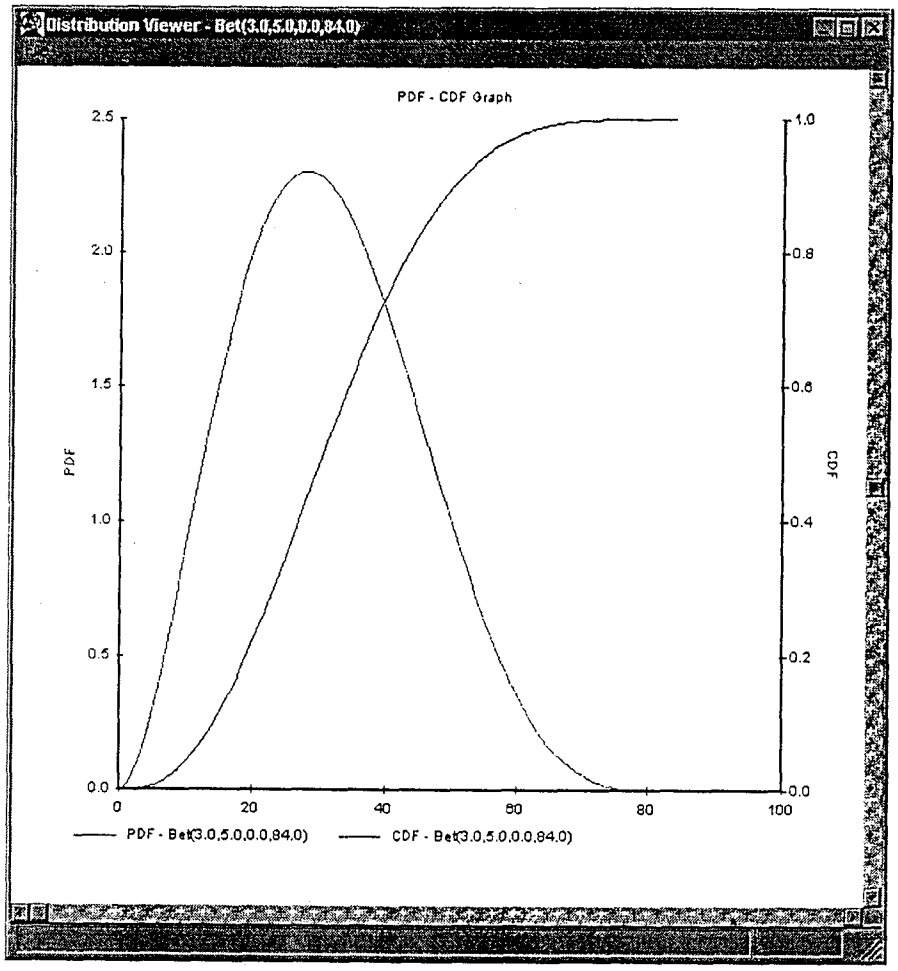


Figure 9 – 1Hz Random Distribution Functions

The simulation of one PEPC LRU FEP updating one workstation for 30 minutes showed that the 1Hz rate could be maintained without any updates

being processed late. Figure 10 graphically depicts the results of this simulation.

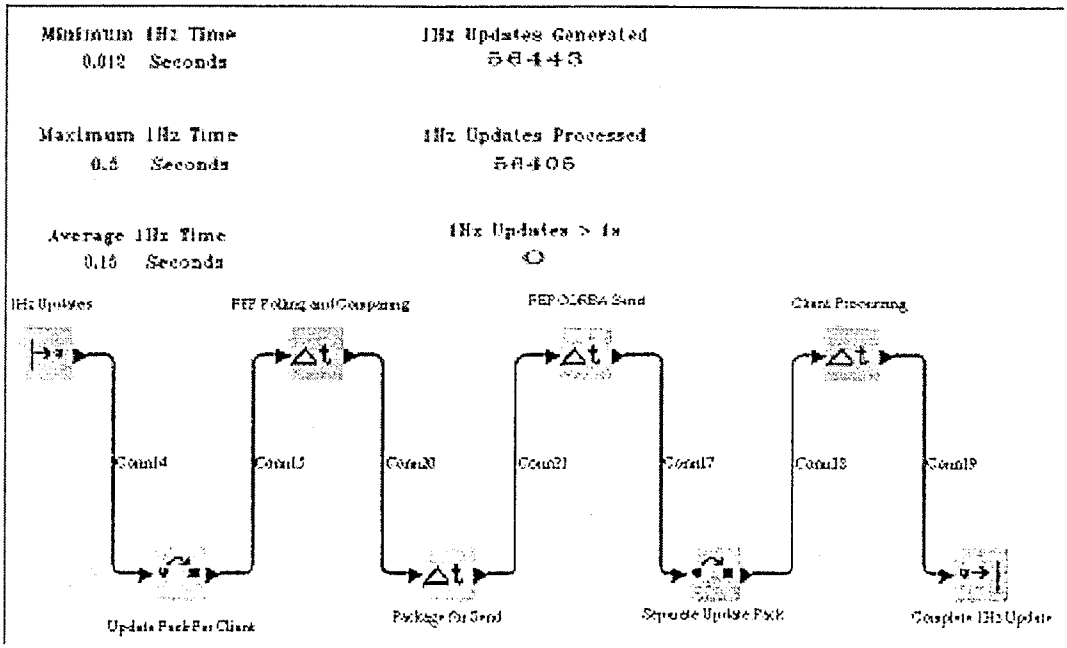


Figure 10 – One PEPC LRU FEP to One Workstation

The simulation of one PEPC LRU FEP updating 14 workstations for 30 minutes showed that nearly one-half of the updates were processed late at the 1Hz rate. Figure 11 graphically depicts the results of this simulation.

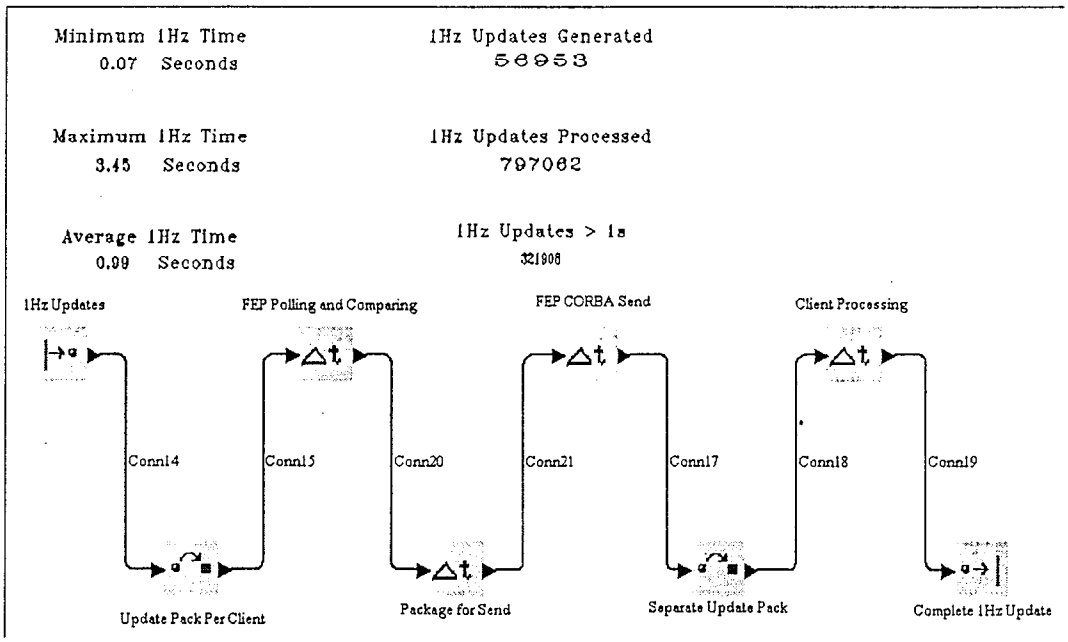


Figure 11 – One PEPC LRU FEP to 14 Workstations

There is only one PEPC supervisory console. It is expected that only one operator at that console will be monitoring the PEPC devices. Thus it is concluded that the status monitor can meet the needs of the PEPC operators.

Simulation Enhancement Recommendations

This section describes enhancements that should be considered in a more detailed study of the status monitor under simulation.

Introduce Resource Contention

Parallel simulation paths should be introduced to model other applications and the operating system consuming computer resources during the simulation runs. By this manner the status monitor simulation path would be forced to act in a more real system-like manner in acquiring shared resources to do its work rather than always receiving those resources when required.

Multiple Update Rates Per FEP

Many of the ICCS FEPs will have status monitor requirements for different poll rates. The current simulations only allow for one poll rate per FEP. The simulations should be modified to more accurately model expected ICCS FEP configurations in terms of the poll rates and update count ranges for each rate that are expected.

Cross Product of FEPs and Clients

In the ICCS FEPs will update multiple clients and clients will receive updates from multiple FEPs. The current simulations only allow for one FEP to update one or more clients. The simulations should be modified to allow the clients to receive updated from one or more FEPs.

Expand FEP CORBA Send Activity

A CORBA send consists of marshalling data, message exchange with receiver, unmarshalling data, receiver acknowledging message receipt, and FEP receiving the acknowledgement. The resource requirement mix changes through these activities whereas in the current simulation it does not because they are lumped into a single activity.

Conclusion

Under the assumptions and distributions of this simulation, the status monitor software can meet the requirements set forth in the SSDR.

Distributing the load over many FEPs and supervisory consoles helps to ensure that no one computer running a status monitor application is overloaded.

Appendix: Acronyms

Acronym	Definition
CDF	Cumulative Distribution Function
FEP	Front End Processor
ICCS	Integrated Computer Control System
LRU	Line Replaceable Unit
NIF	National Ignition Facility
PDF	Probability Distribution Function
PEPC	Plasma Electrode Pockels Cell
SSDR	Subsystem Design Requirement